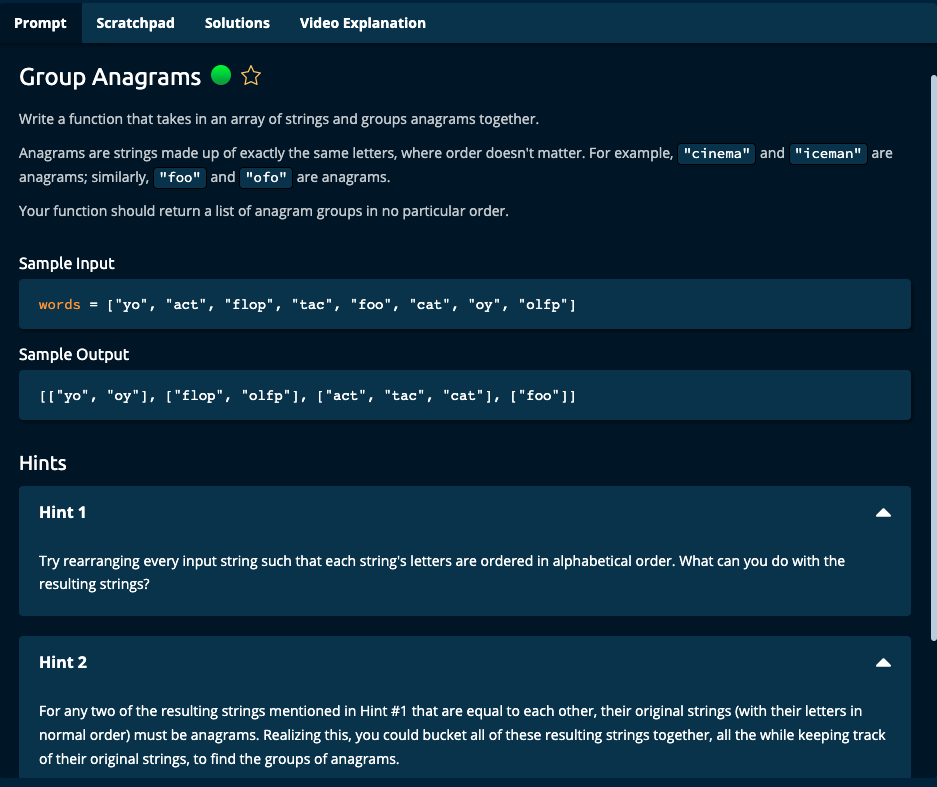
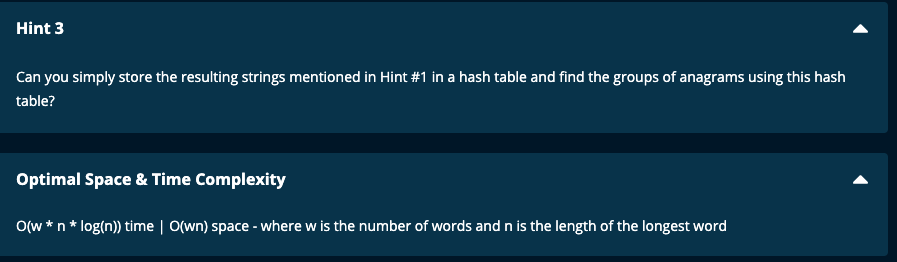
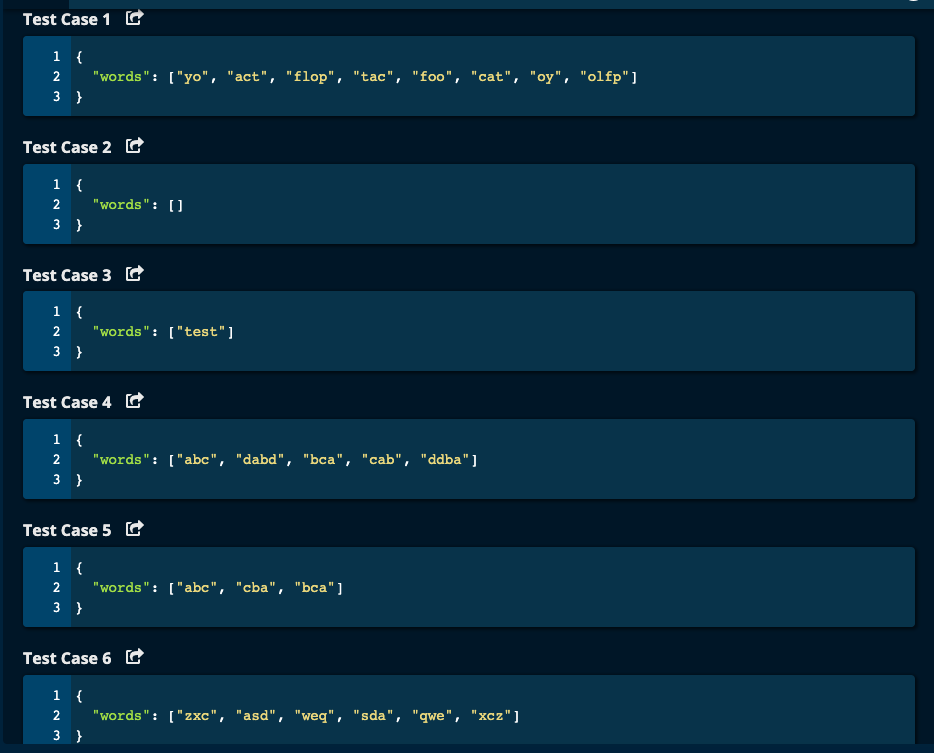
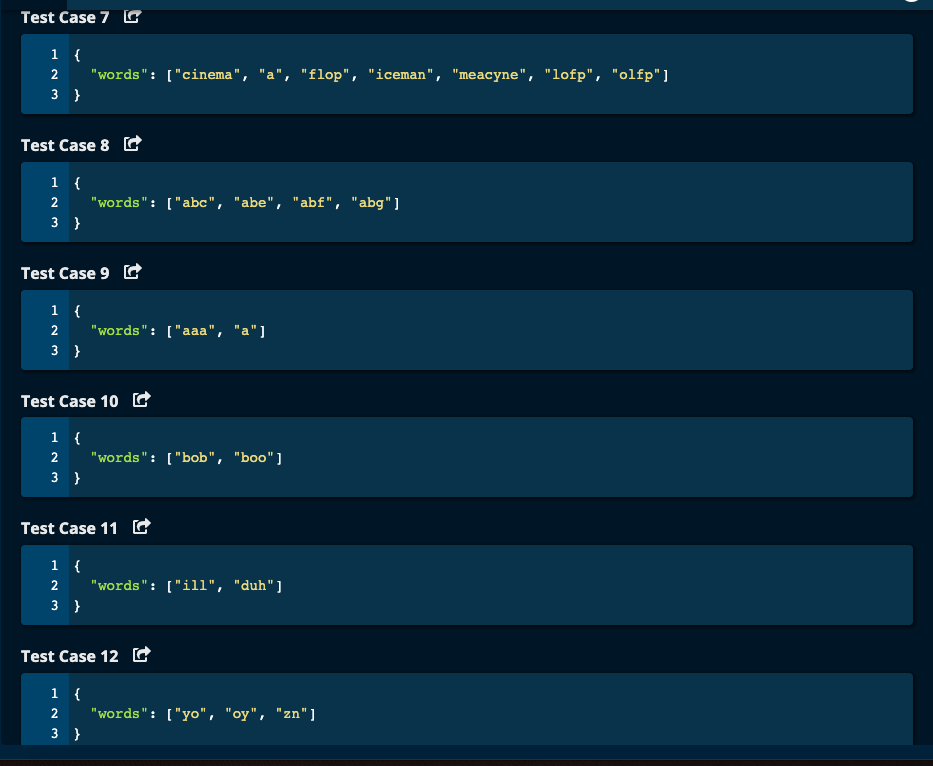
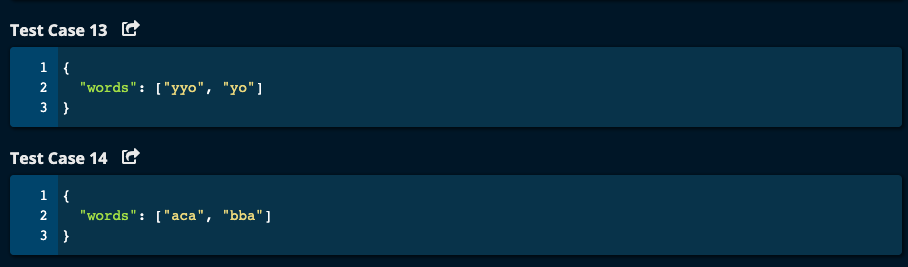
Group Anagram (Medium)











My Solution:

JJ Notes:

1. Initialize a dictionary called adict to an empty one.
2. Traverse through words array. For each word, convert it to a list called wordlist.

Sort this wordlist and make it into a string using join() called sorted\_word.

If sorted\_word is a key in adict, then add this to addict with sorted\_word as key and a list with word as the value.

If sorted\_word is already in adict, then append the word to the value list.

1. Return a list with values from adict.
2. Time Complexity: O(w \* n \* log(n)) where w is the number of words and n is the length of the longest word, since we are sorted each word.

Space Complexity: O(w \* n).

-----------------

def groupAnagrams(words):

adict = {}

for word in words:

sorted\_word = ''.join(sorted(word))

if sorted\_word not in adict.keys():

adict[sorted\_word] = [word]

else:

adict[sorted\_word].append(word)

return list(adict.values())

Algoexpert Solution:

Almost same as my solution.

